Object Oriented Programming

August 29, 2021

Difference between C structures and C++ structures

In C++, struct and class are exactly the same things, except for that struct defaults to public visibility and class defaults to private visibility.

- Member functions inside structure: Structures in C cannot have member functions inside structure but Structures in C++ can have member functions along with data members.

- Direct Initialization: We cannot directly initialize structure data members in C but we can do it in C++.

Difference between C structures and C++ structures

- Using struct keyword: In C, we need to use struct to declare a struct variable. In C++, struct is not necessary. For example, let there be a structure for Record. In C, we must use "struct Record" for Record variables. In C++, we need not use struct and using 'Record' only would work.

- Static Members: C structures cannot have static members but is allowed in C++.

- Constructor creation in structure: Structures in C cannot have constructor inside structure but Structures in C++ can have Constructor creation.

Difference between C structures and C++ structures

- Data Hiding: C structures do not allow concept of Data hiding but is permitted in C++ as C++ is an object oriented language whereas C is not.

- Access Modifiers: C structures do not have access modifiers as these modifiers are not supported by the language. C++ structures can have this concept as it is inbuilt in the language.

Object Oriented Programming

August 29, 2021

Classes and Objects

- Class: A class in C++ is the building block, that leads to Object-Oriented programming. It is a user-defined data type, which holds its own data members and member functions, which can be accessed and used by creating an instance of that class.

- For Example: Consider the Class of Student. There may be many Student from different places but all of them will share some common properties like all of them will have name, register number, department etc. So here, Student is the class and name, register number, department are their properties.

Classes and Objects

- A Class is a user defined data-type which has data members and member functions.
- Data members are the data variables and member functions are the functions used to manipulate these variables. These data members and member functions defines the properties and behavior of the objects.
- In the above example of class Student, the data member will be Name, Register Number and member functions can be Calculate CGP, Display Student details etc.

Classes and Objects

- An Object is an instance of a Class. When a class is defined, no memory is allocated but when it is instantiated (i.e. an object is created) memory is allocated.

- A class is defined in C++ using keyword class followed by the name of class. The body of class is defined inside the curly brackets and terminated by a semicolon at the end.

- class ClassName
  {
  Access Specifier: // can be public, private or protected
  Data Members; // variables to be used
  Member Functions() {} //Methods to access the data members
  };

Procedure Oriented Programming

- Procedure Oriented Programming can be defined as a programming model which is derived from structured programming, based upon the concept of calling procedure.
- Procedures, also known as routines, subroutines or functions, simply consist of a series of computational steps to be carried out.
- During a program's execution, any given procedure might be called at any point, including by other procedures or itself.

Object Oriented Programming

- Object oriented programming can be defined as a programming model which is based upon the concept of objects.
- Objects contain data in the form of attributes and code in the form of methods.
- In object oriented programming, computer programs are designed using the concept of objects that interact with real world. Object oriented programming languages are various but the most popular ones are class-based, meaning that objects are instances of classes, which also determine their types.

Difference between Procedure Oriented Programming and Object Oriented Programming

Procedure Oriented Programming	**Object Oriented Programming**
In Procedure Oriented Programming, program is divided into small parts called functions.	In object oriented programming, program is divided into small parts called objects.
Procedure Oriented Programming follows top down approach.	Object oriented programming follows bottom up approach.
There is no access specifier in Procedure Oriented Programming.	Object oriented programming have access specifiers like private, public, protected etc.
Adding new data and function is not easy.	Adding new data and function is easy.

Difference between POP and OOP

Procedure Oriented Programming does not have any proper way for hiding data so it is less secure.	Object oriented programming provides data hiding so it is more secure.
In Procedure Oriented Programming, overloading is not possible.	Overloading is possible in object oriented programming.
In Procedure Oriented Programming, function is more important than data.	In object oriented programming, data is more important than function.
Procedure Oriented Programming is based on unreal world.	Object oriented programming is based on real world.
Examples: C, FORTRAN, Pascal, Basic etc.	Examples: C++, Java, Python, C# etc.

Difference between Top-Down Approach and Bottom-Up Approach in Programming

Top-Down Approach	**Bottom-Up Approach**
In this approach We focus on breaking up the problem into smaller parts.	In bottom up approach, we solve smaller problems and integrate it as whole and complete the solution.
Mainly used by structured programming language such as COBOL, Fortan, C etc.	Mainly used by object oriented programming language such as C++, C#, Python.
Each part is programmed separately therefore contain redundancy.	Redundancy is minimized by using data encapsulation and data hiding.
In this the communications is less among modules.	In this module must have communication.

Object Oriented Programming

August 29, 2021

C++ Access Specifiers

The **public** keyword is an access specifier. Access specifiers define how the members (attributes and methods) of a class can be accessed. In the example above, the members are public - which means that they can be accessed and modified from outside the code.
In C++, there are three access specifiers:

- public - members are accessible from outside the class
- private - members cannot be accessed (or viewed) from outside the class
- protected - members cannot be accessed from outside the class, however, they can be accessed in inherited classes.

C++ Access Specifiers

Public: All the class members declared under public will be available to everyone. The data members and member functions declared public can be accessed by other classes too. The public members of a class can be accessed from anywhere in the program using the direct member access operator (.) with the object of that class.

C++ Access Specifiers

Private: The class members declared as private can be accessed only by the functions inside the class. They are not allowed to be accessed directly by any object or function outside the class. Only the member functions or the friend functions are allowed to access the private data members of a class.

C++ Access Specifiers

Protected: Protected access modifier is similar to that of private access modifiers, the difference is that the class member declared as Protected are inaccessible outside the class but they can be accessed by any sub-class(derived class) of that class.

Object Oriented Programming

August 29, 2021

Constructors in C++

A constructor is a member function of a class which initializes objects of a class. In C++, Constructor is automatically called when object(instance of class) create. It is special member function of the class. There are 3 types of constructors:

- Default constructors
- Parametrized constructors
- Copy constructors

Default constructors

- Default constructor is the constructor which doesn't take any argument. It has no parameters.
- Even if we do not define any constructor explicitly, the compiler will automatically provide a default constructor implicitly.

Parameterized Constructors

- It is possible to pass arguments to constructors. Typically, these arguments help initialize an object when it is created. To create a parameterized constructor, simply add parameters to it the way you would to any other function. When you define the constructor's body, use the parameters to initialize the object.

Copy Constructor

- Copy Constructor is a type of constructor which is used to create a copy of an already existing object of a class type
- A copy constructor is a member function which initializes an object using another object of the same class.

Destructors in C++

- Destructor is a member function which destructs or deletes an object.
- Destructors have same name as the class preceded by a tilde (~). Destructors don't take any argument and don't return anything

When is destructor called?

A destructor function is called automatically when the object goes out of scope:

- the function ends
- the program ends
- a block containing local variables ends
- a delete operator is called

Destructors in C++

Can there be more than one destructor in a class?

- No, there can only one destructor in a class with classname preceded by (~), no parameters and no return type.

When do we need to write a user-defined destructor?

- If we do not write our own destructor in class, compiler creates a default destructor for us.
- The default destructor works fine unless we have dynamically allocated memory or pointer in class.
- When a class contains a pointer to memory allocated in class, we should write a destructor to release memory before the class instance is destroyed.

Object Oriented Programming

August 29, 2021

C++ Overloading

If we create two or more members having the same name but different in number or type of parameter, it is known as C++ overloading. Types of overloading in C++ are:

- Function overloading
- Operator overloading

C++ Function Overloading

- Function Overloading is defined as the process of having two or more function with the same name, but different in parameters is known as function overloading in C++.
- In function overloading, the function is redefined by using either different types of arguments or a different number of arguments. It is only through these differences compiler can differentiate between the functions.

C++ Operators Overloading

- Operator overloading is used to overload or redefines most of the operators available in C++.
- It is used to perform the operation on the user-defined data type.
- For example, C++ provides the ability to add the variables of the user-defined data type that is applied to the built-in data types.

Operator that cannot be overloaded are as follows:

- Scope operator (::)
- Sizeof
- member selector(.)
- member pointer selector(*)
- ternary operator(?:)

Object Oriented Programming

August 29, 2021

C++ Inheritance

- In C++, inheritance is a process in which one object acquires all the properties and behaviors of its parent object automatically. In such way, you can reuse, extend or modify the attributes and behaviors which are defined in other class.

- In C++, the class which inherits the members of another class is called derived class and the class whose members are inherited is called base class. The derived class is the specialized class for the base class.

C++ Inheritance

Advantage of C++ Inheritance

- Code reusability: Now you can reuse the members of your parent class. So, there is no need to define the member again. So less code is required in the class.

Types Of Inheritance: C++ supports five types of inheritance:

- Single inheritance
- Multiple inheritance
- Hierarchical inheritance
- Multilevel inheritance
- Hybrid inheritance

C++ Inheritance

The Syntax of Derived class:

```
class derived_class_name :: visibility-mode base_class_name
{
    // body of the derived class.
}
```

Details:

- derived_class_name: It is the name of the derived class.
- visibility mode: The visibility mode specifies whether the features of the base class are publicly inherited or privately inherited. It can be public or private.
- base_class_name: It is the name of the base class.

C++ Single Inheritance

- Single inheritance is defined as the inheritance in which a derived class is inherited from the only one base class.

- When one class inherits another class, it is known as single level inheritance.

Protected: Access Specifier

- Protected: Protected access modifier is similar to that of private access modifiers, the difference is that the class member declared as Protected are inaccessible outside the class but they can be accessed by any subclass(derived class) of that class.

C++ Multi Level Inheritance Example

- When one class inherits another class which is further inherited by another class, it is known as multi level inheritance in C++.
- Inheritance is transitive so the last derived class acquires all the members of all its base classes.

C++ Multiple Inheritance

- Multiple Inheritance is a feature of C++ where a class can inherit from more than one classes. i.e one sub class is inherited from more than one base classes.

- Multiple inheritance is the process of deriving a new class that inherits the attributes from two or more classes.

C++ Hierarchical Inheritance

- In this type of inheritance, more than one sub class is inherited from a single base class. i.e. more than one derived class is created from a single base class.

- Hierarchical inheritance is defined as the process of deriving more than one class from a base class.

C++ Hybrid Inheritance

- Hybrid Inheritance is implemented by combining more than one type of inheritance. For example: Combining Hierarchical inheritance and Multiple Inheritance.

- Hybrid inheritance is a combination of more than one type of inheritance.

Visibility of Inherited Members

Base class visibility	Derived class visibility		
	Public	Private	Protected
Private	Not Inherited	Not Inherited	Not Inherited
Protected	Protected	Private	Protected
Public	Public	Private	Protected

Figure 1. Visibility of Inherited Members

Object Oriented Programming

September 3, 2021

Access Modifiers in Java

- Default – No keyword required
- Private
- Protected
- Public

Access Modifiers in Java

- Default: The access level of a default modifier is only within the package. It cannot be accessed from outside the package. If you do not specify any access level, it will be the default.
- Default: When no access modifier is specified for a class , method or data member – It is said to be having the default access modifier by default.
- The data members, class or methods which are not declared using any access modifiers i.e. having default access modifier are accessible only within the same package.

Access Modifiers in Java

- Private: The access level of a private modifier is only within the class. It cannot be accessed from outside the class.
- Private: The private access modifier is specified using the keyword private.
- The methods or data members declared as private are accessible only within the class in which they are declared. Any other class of same package will not be able to access these members.

Access Modifiers in Java

- Protected: The access level of a protected modifier is within the package and outside the package through child class. If you do not make the child class, it cannot be accessed from outside the package.
- protected: The protected access modifier is specified using the keyword protected.
- The methods or data members declared as protected are accessible within same package or sub classes in different package.

Access Modifiers in Java

- Public: The access level of a public modifier is everywhere. It can be accessed from within the class, outside the class, within the package and outside the package.
- The public access modifier is specified using the keyword public. The public access modifier has the widest scope among all other access modifiers.
- Classes, methods or data members which are declared as public are accessible from every where in the program. There is no restriction on the scope of a public data members.

Access Modifiers in Java

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

Figure 1. Access Modifiers in Java

Object Oriented Programming

September 6, 2021

Inheritance in Java

- Inheritance is an important pillar of OOP(Object Oriented Programming). It is the mechanism in java by which one class is allow to inherit the features(fields and methods) of another class.

- Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a parent object. It is an important part of OOP

Inheritance in Java

Why use inheritance in java

- For Code Reusability.
- For Method Overriding

Inheritance in Java

- The keyword used for inheritance is extends.
- class derived-class extends base-class
  {
  //methods and fields
  }

Types of inheritance in java

On the basis of class, there can be three types of inheritance in java:

- Single Inheritance
- Multilevel Inheritance
- Hierarchical Inheritance

Types of inheritance in java

- Single Inheritance : In single inheritance, subclasses inherit the features of one superclass.
- Multilevel Inheritance : In Multilevel Inheritance, a derived class will be inheriting a base class and as well as the derived class also act as the base class to other class.
- In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one sub class.In below image, the class A serves as a base class for the derived class B,C and D.

Types of inheritance in java

- Multiple Inheritance (Through Interfaces) : In Multiple inheritance ,one class can have more than one superclass and inherit features from all parent classes.
- Please note that Java does not support multiple inheritance with classes.
- In java, we can achieve multiple inheritance only through Interfaces.

Types of inheritance in java

- Hybrid Inheritance(Through Interfaces) : It is a mix of two or more of the above types of inheritance.
- The hybrid inheritance is also not possible with classes.
- In java, we can achieve hybrid inheritance only through Interfaces.

Object Oriented Programming

September 7, 2021

Method Overriding in Java

- Overriding is a feature that allows a subclass or child class to provide a specific implementation of a method that is already provided by one of its super-classes or parent classes.

- When a method in a subclass has the same name, same parameters or signature and same return type(or sub-type) as a method in its super-class, then the method in the subclass is said to override the method in the super-class.

Rules for Java Method Overriding

- The method must have the same name as in the parent class
- The method must have the same parameter as in the parent class.
- There must be an IS-A relationship (inheritance).

Method Overriding in Java

- Method overriding is one of the way by which java achieve Run Time Polymorphism.
- The version of a method that is executed will be determined by the object that is used to invoke it.
- If an object of a parent class is used to invoke the method, then the version in the parent class will be executed, but if an object of the subclass is used to invoke the method, then the version in the child class will be executed.

Rules for Java Method Overriding

- Overriding and Access-Modifiers : The access modifier for an overriding method can allow more, but not less, access than the overridden method. For example, a protected instance method in the super-class can be made public, but not private, in the subclass.
- Static methods can not be overridden
- Private methods can not be overridden
- The overriding method must have same return type

Difference between Method Overloading and Method Overriding in Java

Method Overloading	Method Overriding
Method overloading is a compile time polymorphism	Method overriding is a run time polymorphism.
It is occur within the class.	While it is performed in two classes with inheritance relationship.
Method overloading may or may not require inheritance.	While method overriding always needs inheritance.
In this, methods must have same name and different signature.	While in this, methods must have same name and same signature.
Method overloading is used to increase the readability of the program.	Method overriding is used to provide the specific implementation of the method that is already provided by its super class.

Object Oriented Programming

February 6, 2020

Exception Handling in Java

- The Exception Handling in Java is one of the powerful mechanism to handle the runtime errors so that normal flow of the application can be maintained.
- Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IOException, SQLException, RemoteException, etc.
- An exception is an unwanted or unexpected event, which occurs during the execution of a program i.e at run time, that disrupts the normal flow of the program's instructions.

Advantage of Exception Handling

- The core advantage of exception handling is to maintain the normal flow of the application. An exception normally disrupts the normal flow of the application that is why we use exception handling.

- Suppose there are 20 statements in your program and there occurs an exception at statement 12, the rest of the code will not be executed i.e. statement 13 and 20 will not be executed. If we perform exception handling, the rest of the statement will be executed. That is why we use exception handling in Java.

Java Exception Keywords

- try: The "try" keyword is used to specify a block where we should place exception code. The try block must be followed by either catch or finally. It means, we can't use try block alone.
- catch: The "catch" block is used to handle the exception. It must be preceded by try block which means we can't use catch block alone. It can be followed by finally block later.
- finally: The "finally" block is used to execute the important code of the program. It is executed whether an exception is handled or not.
- throw: The "throw" keyword is used to throw an exception.

Object Oriented Programming

September 21, 2021

Interfaces in Java

- An interface is a reference type in Java. It is similar to class.
- It is a collection of abstract methods. A class implements an interface, thereby inheriting the abstract methods of the interface.
- Along with abstract methods, an interface may also contain constants, default methods, static methods.
- All the methods of the interface need to be defined in the class.

An interface is similar to a class in the following ways

- An interface can contain any number of methods.
- An interface is written in a file with a .java extension, with the name of the interface matching the name of the file.

An interface is different from a class in several ways, including

- You cannot instantiate an interface.
- An interface does not contain any constructors.
- All of the methods in an interface are abstract.
- An interface is not extended by a class; it is implemented by a class.
- An interface can extend multiple interfaces.